

## X. Cluster Computing for Embedded/Real-Time Systems

Daniel S. Katz, Jet Propulsion Laboratory, California Institute of Technology,  
Pasadena, CA

Jeremy Kepner, MIT Lincoln Laboratory, Lexington, MA

### X.1 Introduction

Embedded and real-time systems, like other computing systems, seek to maximize computing power for a given price, and thus can significantly benefit from the advancing capabilities of cluster computing. In addition to \$/MFlops, embedded and real-time systems often have severe constraints on size, weight and power as well as latency and reliability. Satisfying these constraints is usually achieved by reducing components such as disk drives, memory and access ports. These additional constraints have traditionally led to more customized solutions within a limited marketplace. More recently, embedded computer vendors have adopted the practice of using clusters of mainstream RISC processors configured in rack mounted cabinets. In addition, many cluster vendors are adopting their rack mounted systems to more compact environments. Reliability constraints in these markets have traditionally been handled by traditional fault avoidance (e.g. applying higher quality and more expensive fabrication procedures on older designs) and fault tolerance (e.g. replication) techniques. While effective, these techniques tend to work against the goals (cost, power, size and weight) of embedded real time systems.

This section of the paper seeks to provide an overview of the cluster computing technology found in the embedded computing world. We begin with a brief description of some of the major applications that use embedded computers. The next two sections present hardware issues (processors, interconnects and form factors) and software issues (operating systems and middleware). Finally, we conclude with a discussion of the current trend towards embedded systems looking more like clusters and clusters looking more like embedded systems.

### X.2 Applications

In this section we briefly discuss three application areas that are driving the development of embedded clusters.

#### X.2.1 Space Applications

One area where embedded clusters are being examined is in space. Spaceborne instruments are providing data at ever increasing rates, far faster than is feasible to send the data to Earth, and the rate of growth of the data from the instruments is also faster than the rate of growth of bandwidth to Earth, so this problem will only get worse in the future. One obvious answer to this is to process the data where it is collected, and to only return the results of the analysis to Earth, rather than the raw data. (A similar argument can be made regarding the signal delay between Earth and space when considering autonomous missions, leading to the same answer: placing the computing where the

decisions must be made.)

Traditionally, very little data analysis has been done in space, and what has been done has relied on radiation-hardened processors. These processors are quite old by the time they complete the radiation-hardening process, and do not solve the problem of bandwidth limitation. What is needed is an embedded cluster of COTS (Commercial-Off-The-Shelf) processors, where the processors can be selected and placed in the system shortly before mission launch. In as much as COTS processors are not radiation hardened, this requires software that can detect and correct errors caused by the cosmic ray environment found in space. Such a system is being developed by the Jet Propulsion Laboratory under the Remote Exploration and Experimentation Project (<http://www-ree.jpl.nasa.gov/>).

The constraints of the space environment (mass, power, size, resistance to vibration, shock, thermal cycling and natural space radiation) demand work in packaging that is very different than for most ground-based clusters. Additionally, replacement of faulty components is either impossible or extremely expensive, and uploading new or changing existing software is very difficult from a system design as well as operational perspective. Finally, reliability of the software as well as the hardware/system is a significant concern due to the difficulty in validating computational results, and the potential impact of erroneous behavior with respect to decision making and scientific data analysis. Thus, a different set of requirements exists for spaceborne computers.

### X.2.2 Signal Processing Applications

The signal processing applications that embedded systems serve are often highly data parallel and naturally lend themselves to the kind of coarse grained parallelism ideal to clusters. However, the quality of service (QoS) and latency requirements of real-time systems usually dictate a need for interconnects with both deterministic performance and higher performance than are typically found in clusters. Thus the solutions developed to serve these needs are similar to conventional rack mounted clusters with high performance interconnects, denser packaging and lighter operating systems.

### X.2.3 Telecommunications Applications

In telecommunications, reliability and real-time response have always been critical factors. Early electronic switching systems used specially designed fault-tolerant computers such as the AT&T 3b2. With the current explosion in network servers, in order to achieve acceptable cost it is necessary to use the types of standard commercial machines and operating systems discussed in this section. This industry is currently experimenting with commodity cluster systems to solve the problem of meeting their ultra-high reliability and real-time response constraints while reducing costs.

## X.3 Hardware

Form factor is a critical factor in embedded computing. Thus compute density (GFlops per cubic foot and GFlops per watt) is often as or more important than aggregate processing power. Typically, embedded system vendors are able to achieve roughly a factor of 10 increase in compute density over conventional systems. These gains are achieved by constructing boards containing multiple nodes (typically 2 or 4). Each node consists of a low power processor (e.g., Motorola PowerPC) with a limited amount of memory (e.g. 128 MBytes). In addition, there are no local disks and access to the node is limited to the interconnect, which may be custom or commodity (e.g. Myrinet), and will have been packaged to minimize size and power consumption. These various design tradeoffs allow embedded vendors to fit nearly 100 processing nodes in a volume that can fit underneath a typical office desk.

As mentioned previously, many embedded systems need to withstand much more severe conditions than standard clusters. These systems may be used in the aerospace or military industries, leading to requirements on tolerance to shock, vibration, radiation, thermal conditions, etc. While many of today's commercial components can handle these conditions, they are not packaged to do so, as this increases cost and is not needed by most ordinary users. Thus, for this niche market, different vendors have sprung up to package standard commercial parts with more consideration of these concerns.

There are a variety of vendors that manufacture systems along the above lines. Mercury, CSPI and Sky are three of the more popular systems. Some of the general capabilities are shown below. In addition, for comparison, a cluster vendor (AltaTech) is also shown.

Vendor	CPU	Interconnect	OS	CPU/ft <sup>3</sup>
Mercury	PowerPC	Raceway	MCOS	~10
CSPI	PowerPC	Myrinet	VxWorks	~10
Sky	PowerPC	Sky Channel	SKYmpx	~10
Alta	Intel/Alpha	Ethernet/Myrinet	Linux	~1

In addition to these vendors that specialize in embedded systems, a number of other companies build embedded systems, both parallel and distributed for their customers. These vendors may take systems from the standard vendors listed above and ruggedize and/or repackage them, and they include many US defense contractors (Lockheed, Honeywell, General Dynamics, etc.)

Many embedded systems are also targeted for real-time applications with extremely low latency requirements (e.g., radar signal processing). To achieve these requirements it often necessary to adopt a pipeline architecture with different processing occurring at each stage of the pipeline. Typically, each stages exploits coarse grain parallelism but the 'direction' of this parallelism is along different dimensions of the data at different steps. To fully exploit a parallel computer in such circumstances requires transposing (or "corner turning") the data

between steps. Thus, the interconnects provided by embedded systems often have higher bandwidth and lower latencies than those of shared memory supercomputers, let alone clusters of workstations.

#### X.4 Software

The real-time requirements of embedded systems necessitate special operating systems and middleware to reduce latency and to fully exploit the interconnects and processors. This has resulted in a wide variety of Unix flavored operating systems: VxWorks, Lynx, MCOS, SKYmpx, LinuxRT, IrixRT. Typically, these operating systems trade off memory protection, multi-user and multi-threaded capabilities to get higher performance. Traditionally, VxWorks has been one of the most common in many industries. However, the need for portable software has led many to examine alternatives operating systems for embedded clusters. These include LinuxRT and IrixRT. The advantage of these choices is that software can be developed (at least to some point) on common desktop machines, and easily transferred to the embedded clusters. Lynx and other POSIX-compliant systems are intended to be used similarly, under the assumption that software developed on one POSIX-compliant operating system can be easily ported to another. The primary distinction between these operating systems and VxWorks is that VxWorks does not provide process-based memory protection, which may be important in prevention of fault propagation from one process to another.

One of the most significant positive trends in embedded computing has been the adoption of common middleware libraries to ease portability between systems. The two major areas where this has occurred is in communications and math libraries. In the past, special vendor libraries were required to fully exploit their custom networks, but recently these vendors have adopted MPI and, through careful optimization, are able to achieve similar performance as with their vendor libraries. MPI has been a large step forward for the embedded community, but it does not address all of the communication needs of these systems. This has led to the development of MPI/RT (<http://www.mpirt.org/>), which provides critical features such as quality of service. In addition to MPI/RT, the Data Reorganization Interface (DRI: <http://www.data-re.org/>) has been another standardization effort to provide a common interface to large data movements.

The mathematics libraries developed by embedded vendors are similar to other optimized math libraries in that they provide a variety of standard mathematical operations which have been tuned to a particular processor. The functional focus of the embedded libraries has primarily been on basic signal processing operations (e.g. FFT, FIR filters, linear algebra) for complex floating point data. Because Fortran compilers for these systems are hard to find, these libraries usually only have a C implementation. One additional feature of these libraries has been the ability to pre-allocate memory. For most operations, this eliminates potential latency. Optimized math libraries are critical to achieving real-time performance and thus these libraries are heavily used in embedded real-time software. This heavy reliance can lead to a significant portability bottleneck. Fortunately, one of the most successful efforts of this community has been the adoption of a standardized Vector, Signal and Image Processing

Library (VSIPL: <http://www.vsipl.org/>). In addition to being valuable to embedded systems, this library has a significant potential benefit to the general high performance computing community.

Creating an optimized implementation of a standard can be a significant undertaking for embedded systems vendors. Increasingly, these libraries are being implemented by third party software developers (e.g. MPI Software Technologies, Inc). However, as the embedded community is a niche market, software vendors generally do not have enough demand to optimize their products for the variety of embedded systems, and thus products which can optimize themselves, such as ATLAS and FFTW (as mentioned in section 7 of this white paper) will be increasingly important.

## X.5 Summary

While there are certainly a number of differences between embedded clusters and standard clusters that have hopefully been brought out in this section, there are also a number of similarities, and in many ways, the two types of clusters are converging. Mass market forces and the need for software portability are driving embedded clusters to use similar operating systems, tools, and interconnects as standard clusters. Additionally, as traditional clusters grow in size and complexity, there is a growing need to use denser packaging techniques and higher bandwidth, lower latency interconnects. Additionally, fault-tolerance is becoming more important for standard clusters: first, as they are increasingly accepted into machine rooms and subject to reliability and up-time requirements; and second, as feature sizes and operating voltages are reduced, cosmic-ray upsets will occur more frequently. The only areas which promise to continue to separate the two cluster worlds is the general need for ruggedized packaging for many embedded clusters.